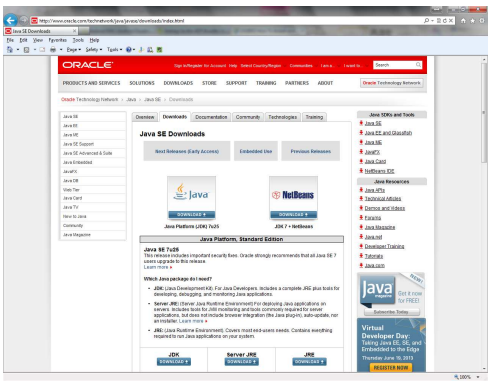
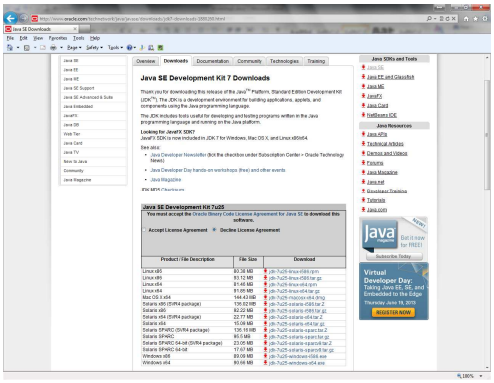
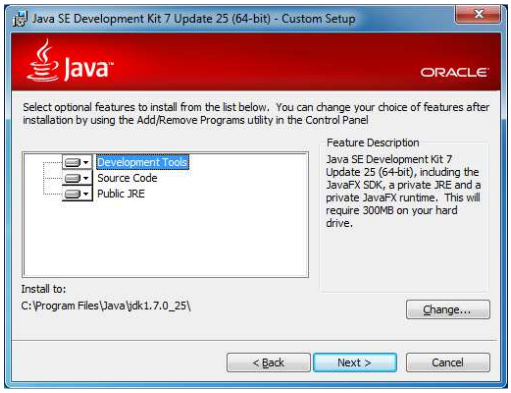
1. Install and Setup ADT

1.1 Installation of Java SE Development Kit (JDK)

1. Download the latest version of Java SE Development Kit (JDK) from the Oracle's Java webpage: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Then click **"Accept License Agreement"** and then select the correct version to download.

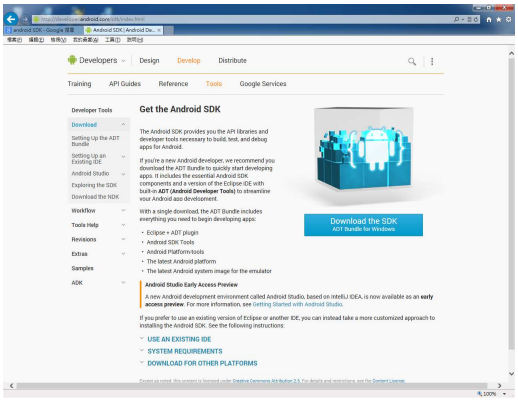
2. Double click the file to start the installation of JDK. Select the component that you want to install (Recommend select all). Select **Change** to modify the path for installing JDK, and then press **Next** to continue.

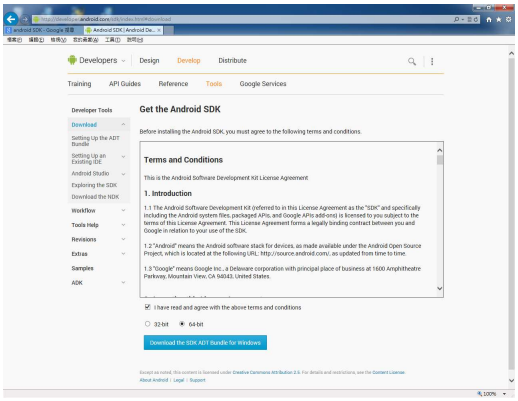


3. Select **Change** to modify the path for installing JRE, and then press **Next** to continue. After the install process finish, click **[Close]** to end.

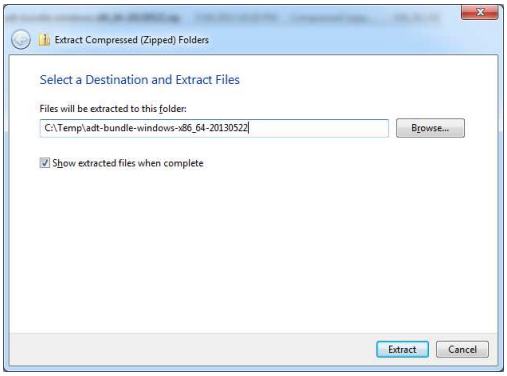
1.2 Installation of Android SDK

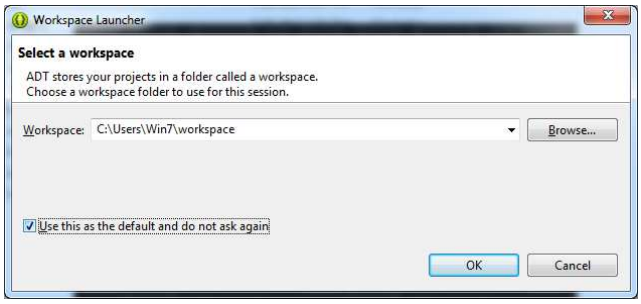
1. Download the latest version of Android SDK from the Android Developer webpage: http://developer.android.com/sdk/index.html. Then click "**I have read and agree with the above terms and conditions**" and then select the correct version to download.



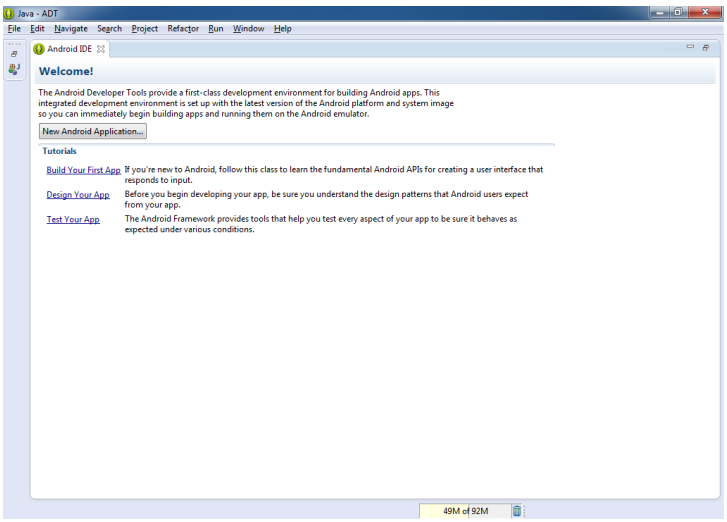


2. Unpack the ZIP file (named adt-bundle-<os_platform>.zip) and save it to an appropriate location, such as a "Development" directory in your home directory. Open the directory (adt-bundle-<os_platform>/eclipse/) and launch Eclipse. Select the workspace when start Eclipse in first time.



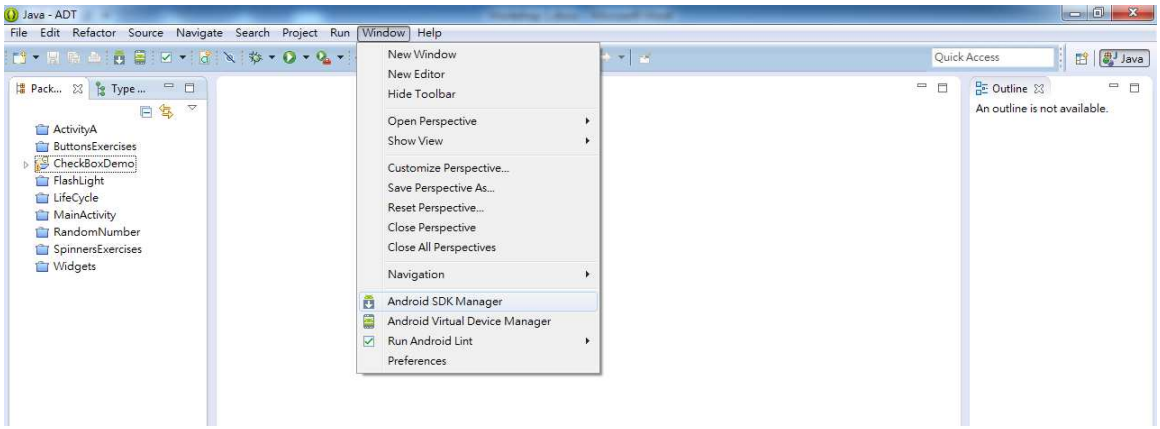


3. The Android Developer Tool is launch.

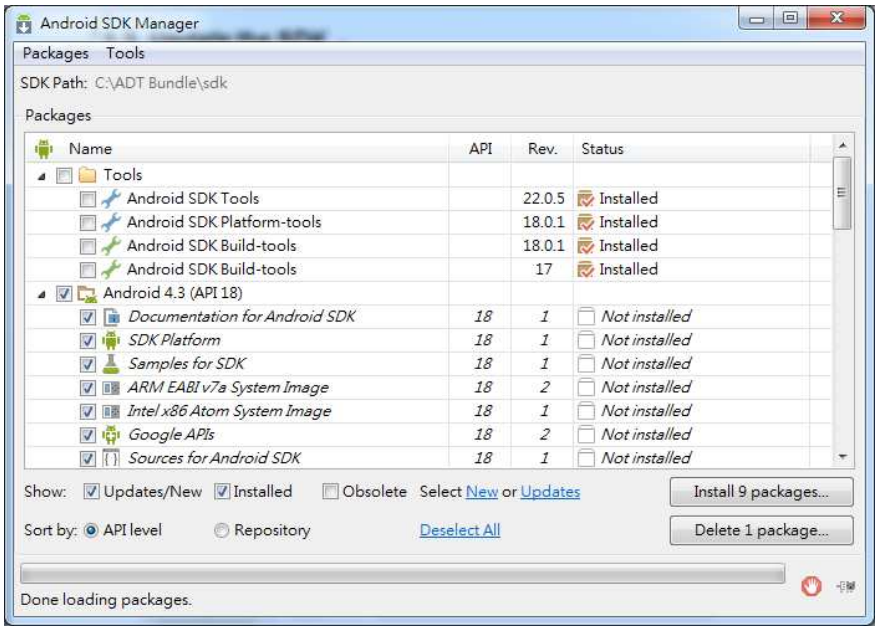


1.3 Update the SDK

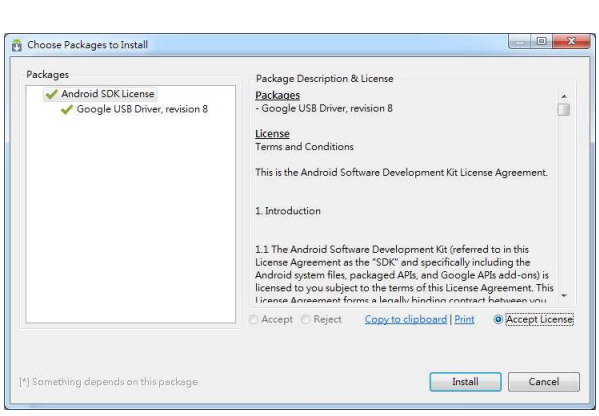
1. Select **Windows ➔ Android SDK Manager**.

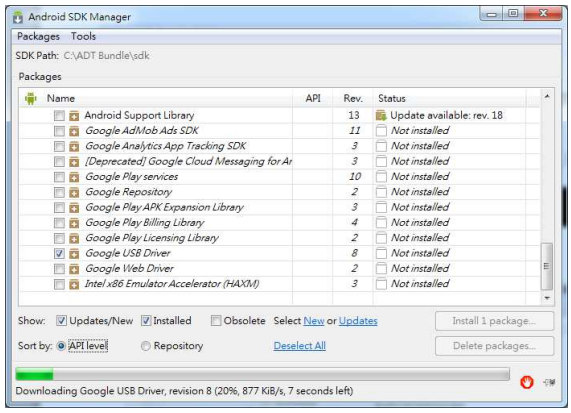


2. Select and install the corresponding components (Recommend all component within Tools, latest Android API and Extras).



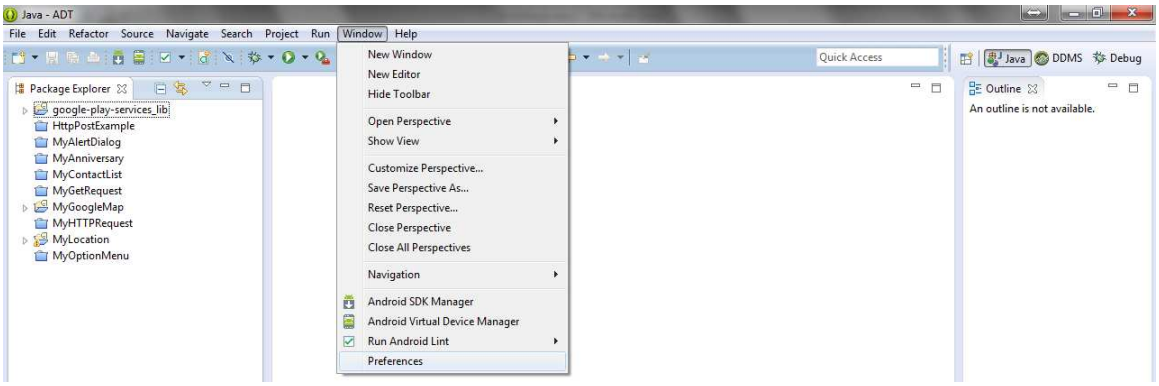
3. Accept the license and press "**Install**" to start, this might take you a hour to complete.



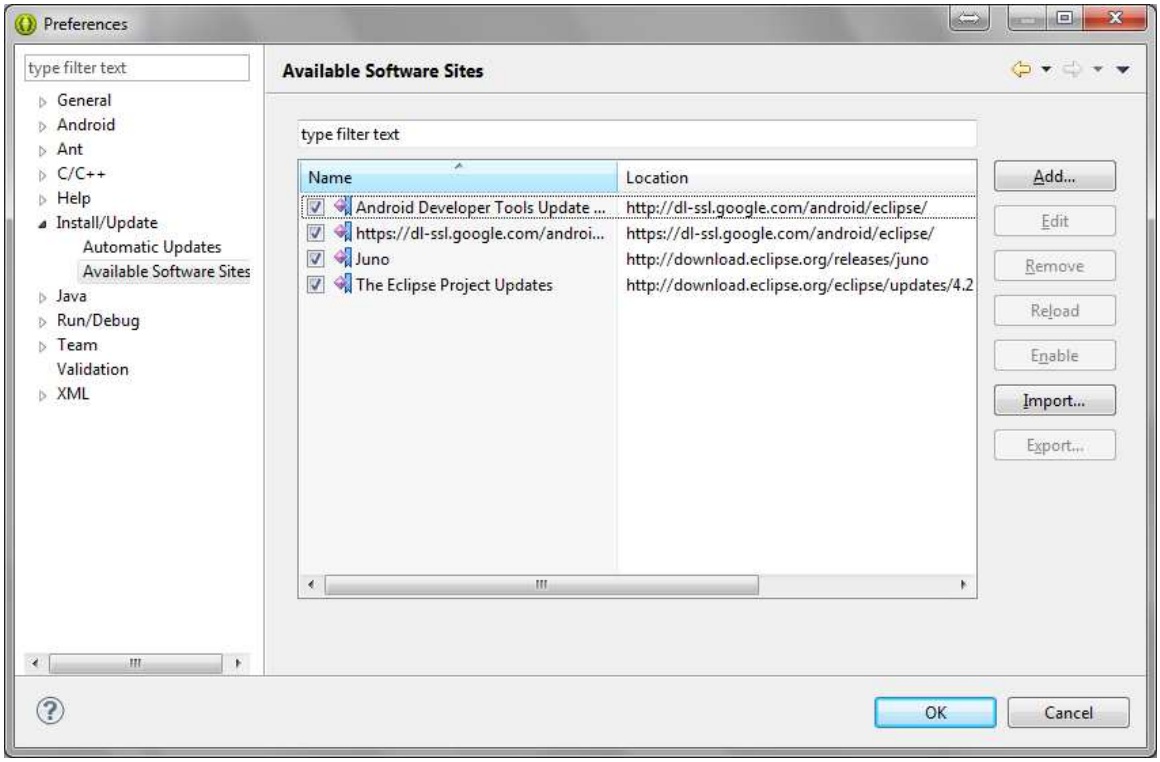


1.4 Update the Eclipse

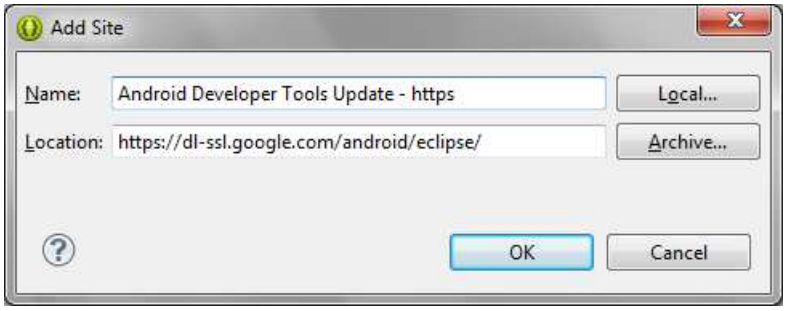
1. Sometime, you are required to update the Eclipse by selecting **Windows ➔ Preferences**.



2. Press the [Add] button from the **Install/Update ➔ Available Software Sites** section in **Preferences** dialog.



3. Set the location as https://dl-ssl.google.com/android/eclipse/, and press **[OK]** to confirm. Then execute **Help ➔ Check for Update** again.

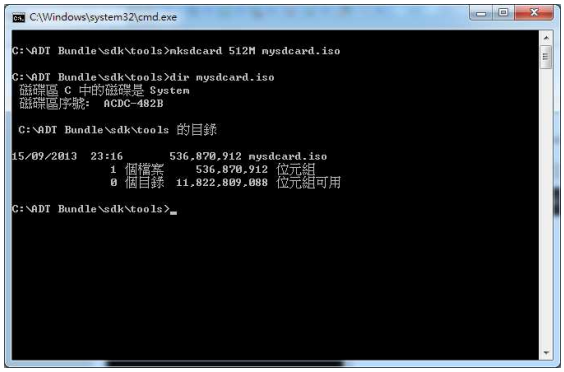


1.5 Create an Android Virtual Device (AVD)

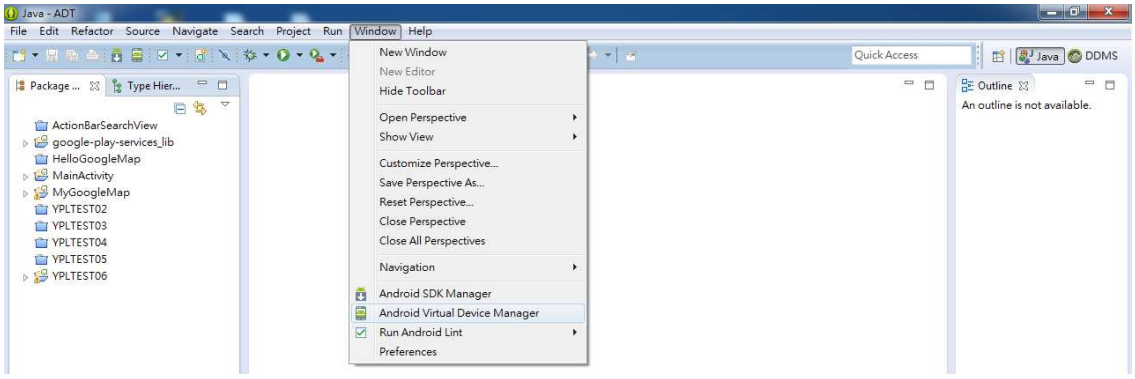
1. To run your app on the emulator you need to first create an Android Virtual Device which is a device configuration for the Android emulator that allows you to model different devices. First you need to create a virtual SD card for your AVD. Go to the SDK Windows tools folder (C:\ [adt-bundle]\ sdk \tools) and execute the following command.

```
mksdcard 512M mysdcard.iso
```

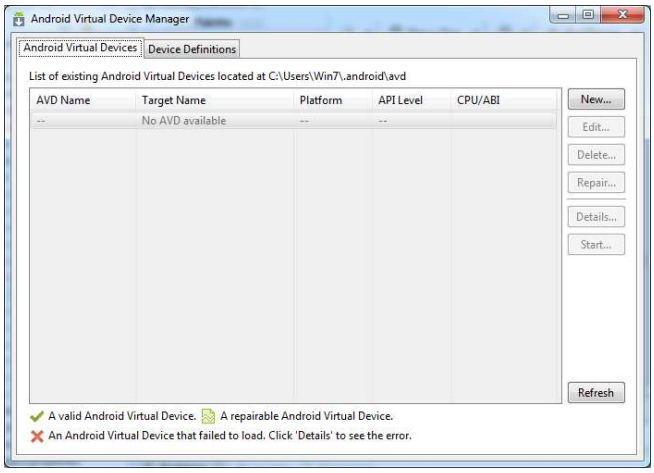
2. After execute the command is DOS Prompt, you can use "DIR" to verify the file.



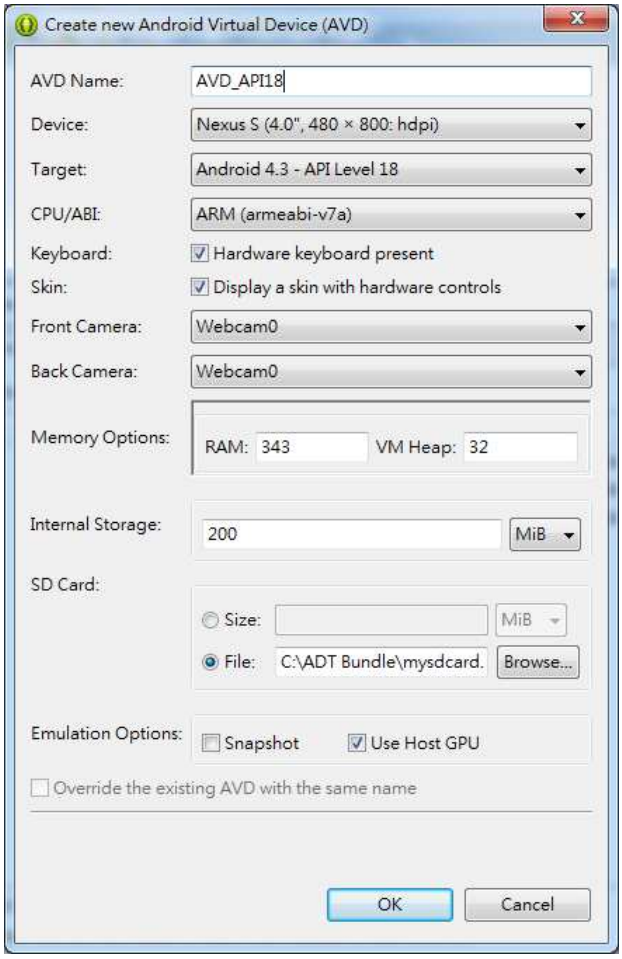
3. Start ADT Bundle, and select **Windows ➔ Android Virtual Device Manager** to launch the Android Virtual Device Manager.



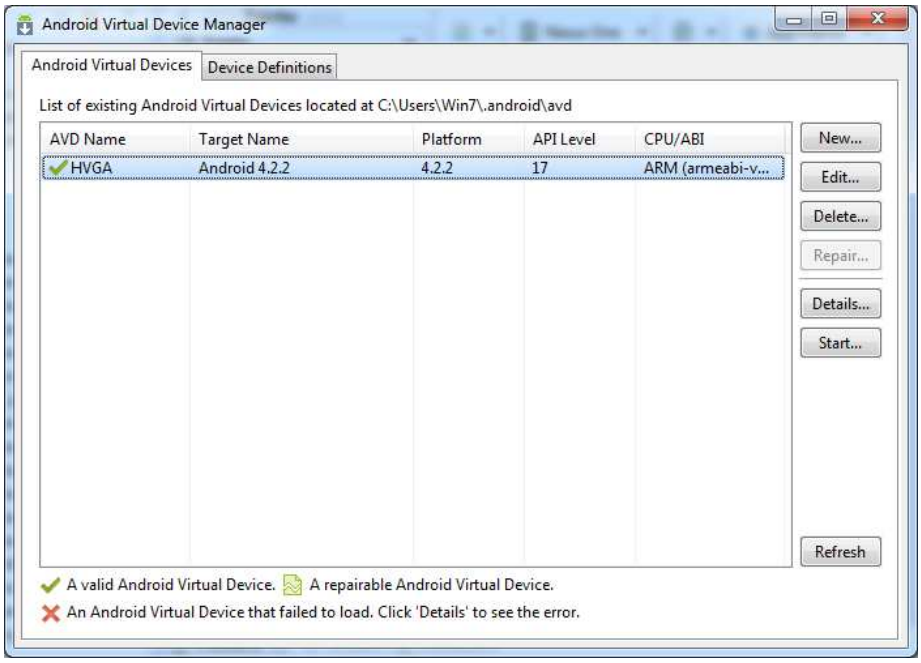
4. Click **New** in the Android Virtual Device Manager panel.



5. Fill in the details for the AVD. Give it a name, device, platform target, an SD card size, and a skin and then press **OK**.



6. The new AVD is listed in the Android Virtual Device Manager, close the device manager



2. Content List

2.1 Retrieve Contact using Content Provider

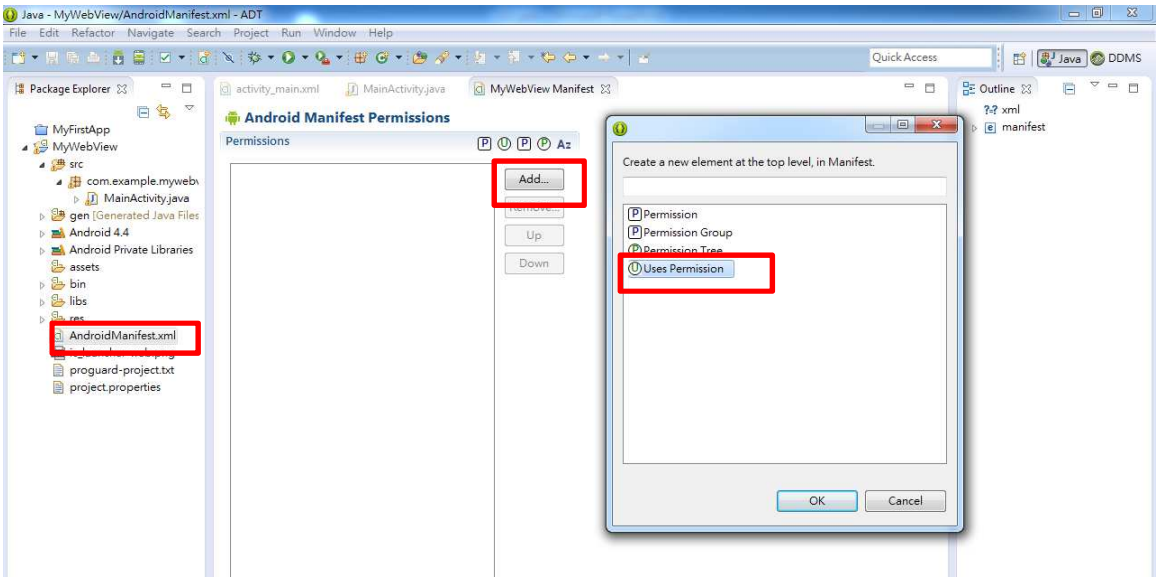
1. Create the Android application with the following attributes.
   - Application Name: **MyContactList**
   - Project Name: **MyContactList**
   - Package Name: **com.example.mycontactlist**

2. Modify the source code for the file "**MainActivity.java**" as follow:

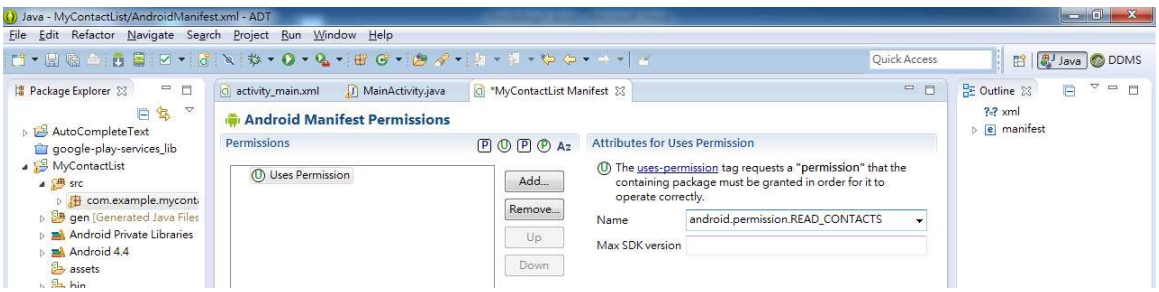
```
package com.example.mycontactlist;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.app.ListActivity;
import android.content.ContentResolver;
import android.provider.ContactsContract;
import android.database.Cursor;
import android.widget.SimpleCursorAdapter;

public class MainActivity extends ListActivity {
    public SimpleCursorAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
//      setContentView(R.layout.activity_main);

        // To retrieve the contact row that contains the user's profile
        ContentResolver contentResolver = getContentResolver();

        // Queries the user dictionary based on selection criteria and returns results
        Cursor cursor = contentResolver.query(
                ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

        // Define a simple Adapter
       adapter = new SimpleCursorAdapter(this,
                        android.R.layout.simple_list_item_1,
```

```
                        cursor,
                        new String[] { ContactsContract.Contacts.DISPLAY_NAME },
                        new int[] { android.R.id.text1 });

        // Create the List Adapter
        setListAdapter(adapter);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

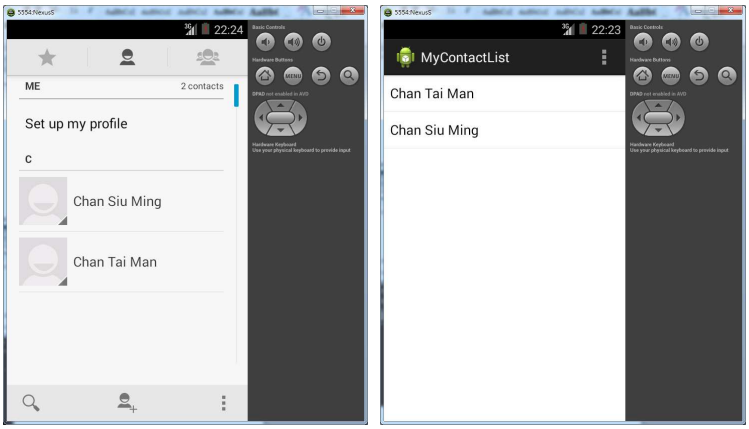
3. Open the Android Manifest file, and press **[Add]** in the "**Permission**" tab. Then select "**Uses Permission**" and press **[OK]**.



4. Set the permission name as "**android.permission.READ_CONTACTS**" and save:

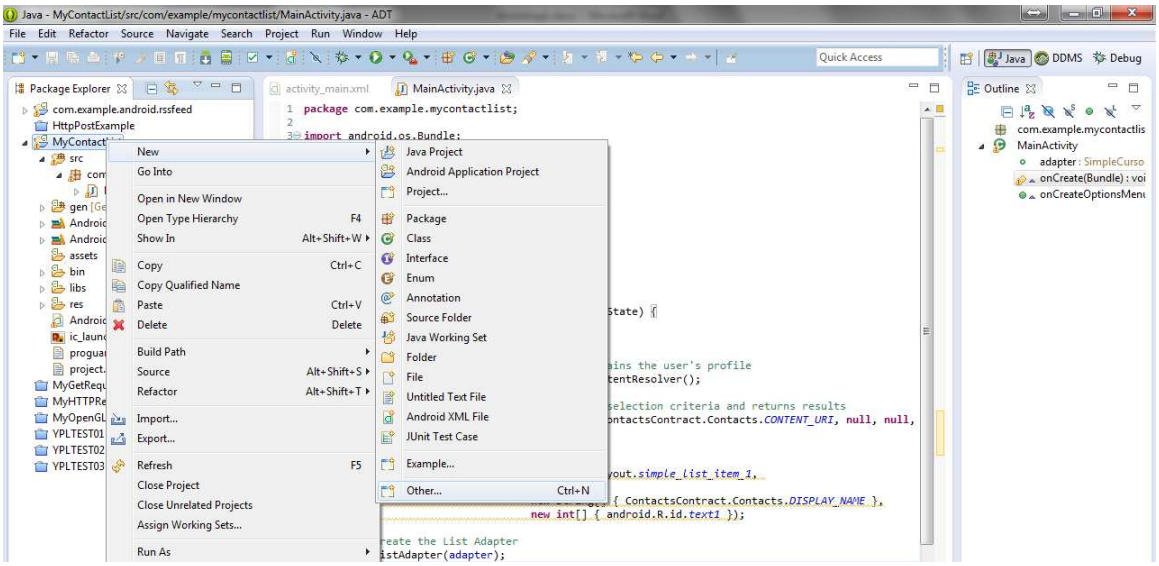


5. Save and execute the app, can you retrieve the existing contact list from your mobile? Remember to create some contact in your emulator if you test this app in AVD.

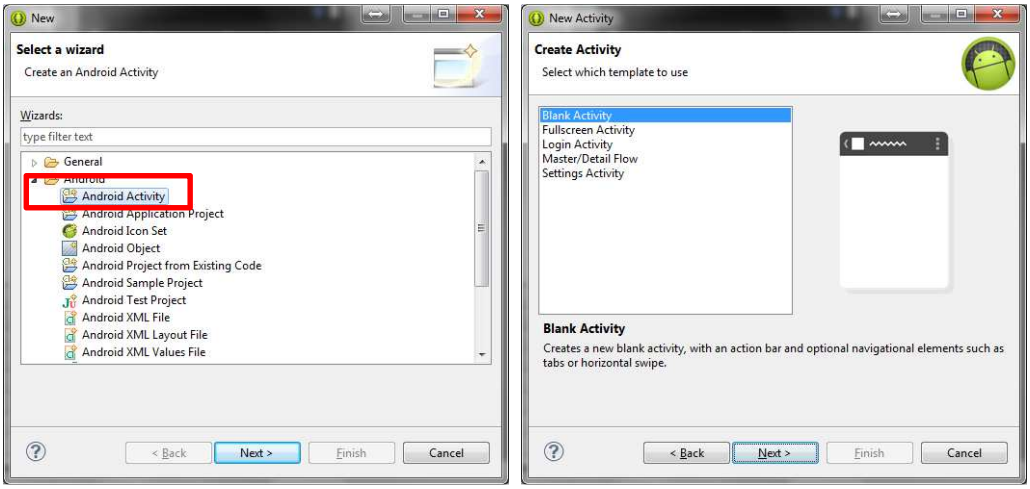


2.2 Display Contact Detail by Intent

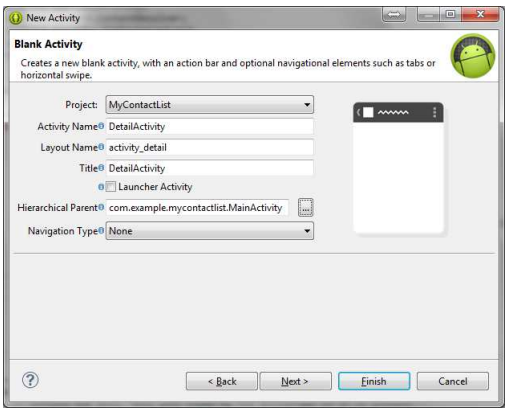
1. Right click the project "**MyContactList**", then select **New ➔ Other…**



2. Select **Android ➔ Android Activity** and press "**Next**" to continue.



3. Input the following attributes for the new activity press **[Finish]** button:
   - Project Name: **MyContactList**
   - Activity Name: **DetailActivity**
   - Layout Name: **activity_detail**
   - Hierarchical Parent: **com.example.mycontactlist.MainActivity**



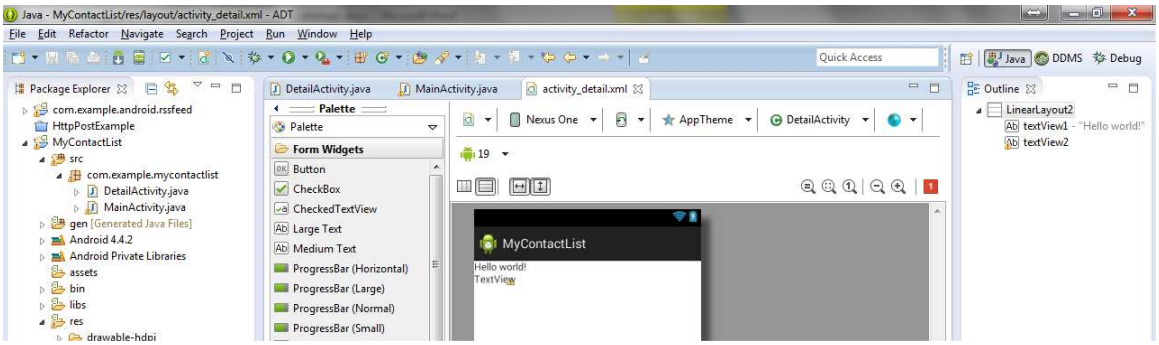
4. In order for the next activity to query the extra data, you should define the key for the intent's extra using a public constant. So add the extra message definition to the top of the **MainActivity** class. Moreover, in order to start an activity, call *startActivity()* and pass it your Intent. The system receives this call and starts an instance of the Activity specified by the Intent modify "**MainActivity.java**" as:

```
package com.example.mycontactlist;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.app.ListActivity;
import android.content.ContentResolver;
import android.provider.ContactsContract;
import android.database.Cursor;
import android.widget.SimpleCursorAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.view.View;
import android.content.Intent;

public class MainActivity extends ListActivity {
    public SimpleCursorAdapter adapter;
    public final static String EXTRA_MESSAGE1 = "MyContactList.cid";
    public final static String EXTRA_MESSAGE2 = "MyContactList.Nickname";
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
//      setContentView(R.layout.activity_main);

        // To retrieve the contact row that contains the user's profile
        ContentResolver contentResolver = getContentResolver();

        // Queries the user dictionary based on selection criteria and returns results
        Cursor cursor = contentResolver.query(
                ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

        // Define a simple Adapter
       adapter = new SimpleCursorAdapter(this,
                        android.R.layout.simple_list_item_1,
                        cursor,
                        new String[] { ContactsContract.Contacts.DISPLAY_NAME },
                        new int[] { android.R.id.text1 });

        // Create the List Adapter
        setListAdapter(adapter);
    }

    // Handle Item click event
    protected void onListItemClick(ListView list, View view, int position, long id){
        // Retrieve the user input
        Cursor cursor = (Cursor) adapter.getItem(position);

        // Get the Cursor Column Index
        int ColumnIndex = cursor.getColumnIndex(ContactsContract.Contacts._ID);

        // Convert the cursor Column Index into string
        String cid = cursor.getString(ColumnIndex);

        // Get the nickname
        String Nickname = (String) ((TextView) view).getText();

        // create an Intent to start an activity called DisplayMessageActivity
```

```
        Intent intent = new Intent(this, DetailActivity.class);

        // An Intent can carry a collection as key-value pairs called extras.
        intent.putExtra(EXTRA_MESSAGE1, cid);
        intent.putExtra(EXTRA_MESSAGE2, Nickname);

        // Switch to another activity
        startActivity(intent);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

5. Drag a TextView to the layout "**activity_detail.xml**".



6. The XML code for the layout should look like:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".DetailActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
</LinearLayout>
```

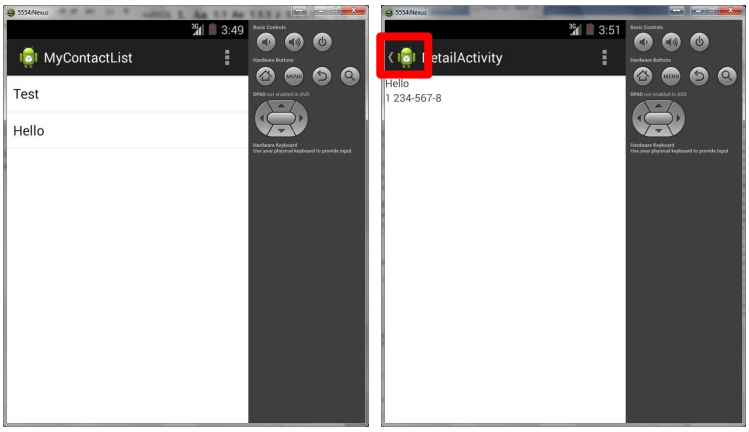
7. Modify "**DetailActivity.java**" as follow:

```
package com.example.mycontactlist;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
import android.provider.ContactsContract;
import android.content.ContentResolver;
import android.content.Intent;
import android.database.Cursor;
import android.widget.TextView;

public class DetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);
        // Show the Up button in the action bar.
        setupActionBar();

        // Get the message from the intent
        Intent intent = getIntent();
        String cid = intent.getStringExtra(MainActivity.EXTRA_MESSAGE1);
        String Nickname = intent.getStringExtra(MainActivity.EXTRA_MESSAGE2);

        // Construct the Content Resolver
        ContentResolver cr = getContentResolver();
```

```
        // Read Contact number of specific contact with help of Content Resolver
        Cursor cursor = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                new String[] { ContactsContract.CommonDataKinds.Phone.NUMBER },
                ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=?",
                new String[] { cid },
                null);

        // move the cursor to first row
        cursor.moveToFirst();

        // Obtain the Telephone Number
        int ColumnIndex =
              cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
        String PhoneNumber = cursor.getString(ColumnIndex);

        // Display Nickname on Screen
        TextView textView1 = (TextView) findViewById(R.id.textView1);
        textView1.setText(Nickname);

        // Display Telephone Number on Screen
        TextView textView2 = (TextView) findViewById(R.id.textView2);
        textView2.setText(PhoneNumber);
    }

    /**
     * Set up the {@link android.app.ActionBar}.
     */
    private void setupActionBar() {
        getActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.detail, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case android.R.id.home:
            // This ID represents the Home or Up button. In the case of this
            // activity, the Up button is shown. Use NavUtils to allow users
            // to navigate up one level in the application structure. For
            // more details, see the Navigation pattern on Android Design:
            //
            // http://developer.android.com/design/patterns/navigation.html#up-vs-back
            //
            NavUtils.navigateUpFromSameTask(this);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

8. Save and execute the app, the contact list is shown on the screen. The detail information will be displayed by pressing the corresponding list. You can press the back button in Action Bar to return to main page.

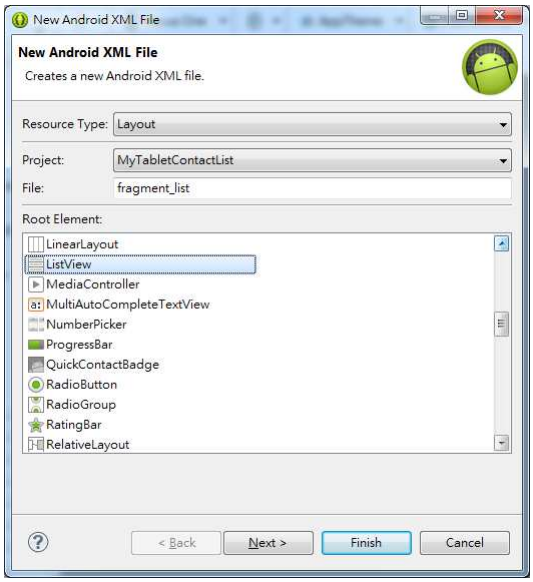


3. Contact List for Tablet

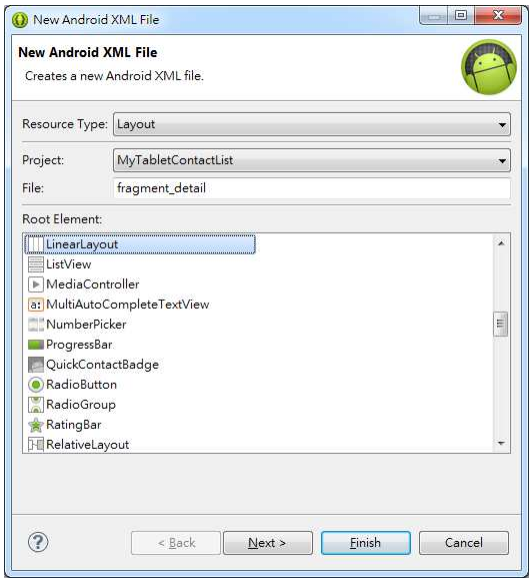
3.1 Duel Pane

1. Create the Android application with the following attributes.
   - Application Name: **MyTabletContactList**
   - Project Name: **MyTabletContactList**
   - Package Name: **com.example.mytabletcontactlist**

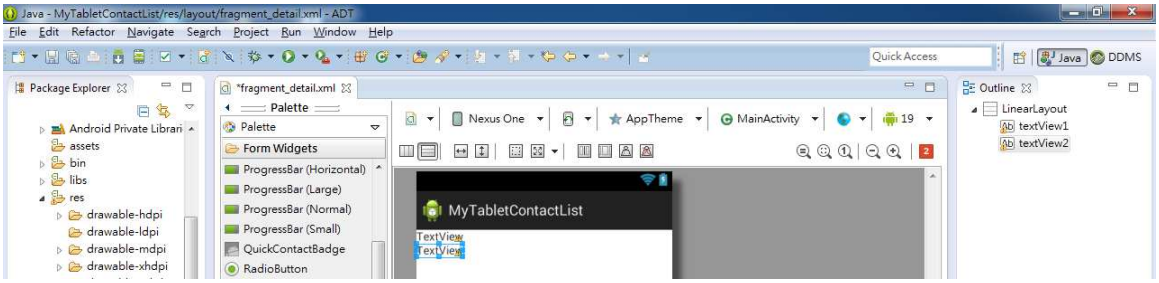
2. Right click the layout "**res/layout**", and select **New ➔ Android XML File**. Input "**fragment_list**" as the file name and select "**ListView**". Then press **[Finish]** button to continue.



3. Right click the layout "**res/layout**", and select **New ➔ Android XML File**. Input "**fragment_detail**" as the file name and select "**LinearLayout**". Then press **[Finish]** button to continue.



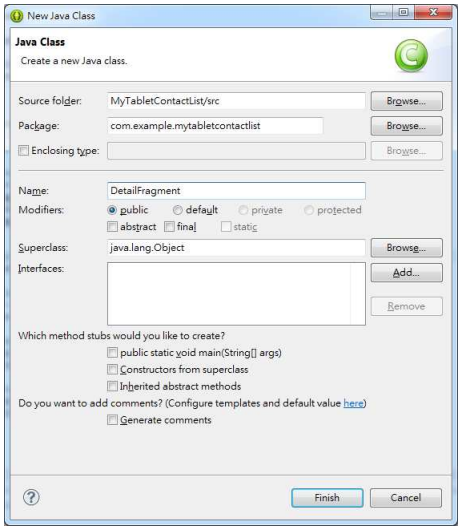
4. Drag two TextView to the layout "**fragment_detail.xml**".



5. The XML code for "**fragment_detail.xml**" will look like:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
</LinearLayout>
```

6. Right click the package **src/com.example.mytabletcontactlist**, and select **New ➔ Class**. Input "**DetailFragment**" as the class name and press **[Finish]** button.

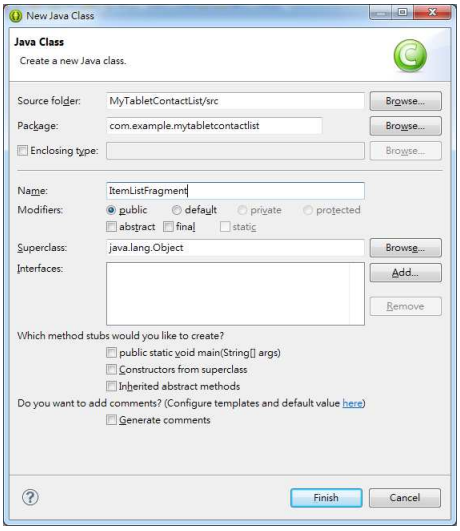


7. Modify "**DetailFragment.java**" as follow:

```
package com.example.mytabletcontactlist;

import android.app.Fragment;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class DetailFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_detail, container, false);
        return view;
    }

    public void setText(String cid, String Nickname) {
        // Construct the Content Resolver
        ContentResolver cr = getActivity().getContentResolver();

        // Read Contact number of specific contact with help of Content Resolver
        Cursor cursor = cr.query(
                ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                new String[] { ContactsContract.CommonDataKinds.Phone.NUMBER },
                ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=?",
                new String[] { cid },
                null );

        // move the cursor to first row
        cursor.moveToFirst();

        // Obtain the Telephone Number
        int ColumnIndex =
            cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
```

```
            String PhoneNumber = cursor.getString(ColumnIndex);

            // Display Nickname on Screen
            TextView textView1 = (TextView) getView().findViewById(R.id.textView1);
            textView1.setText(Nickname);

            // Display Telephone Number on Screen
            TextView textView2 = (TextView) getView().findViewById(R.id.textView2);
            textView2.setText(PhoneNumber);
        }
}
```

8. Right click the package **src/com.example.mytabletcontactlist**, and select **New ➔ Class**. Input "**ItemListFragment**" as the class name and press **[Finish]** button.



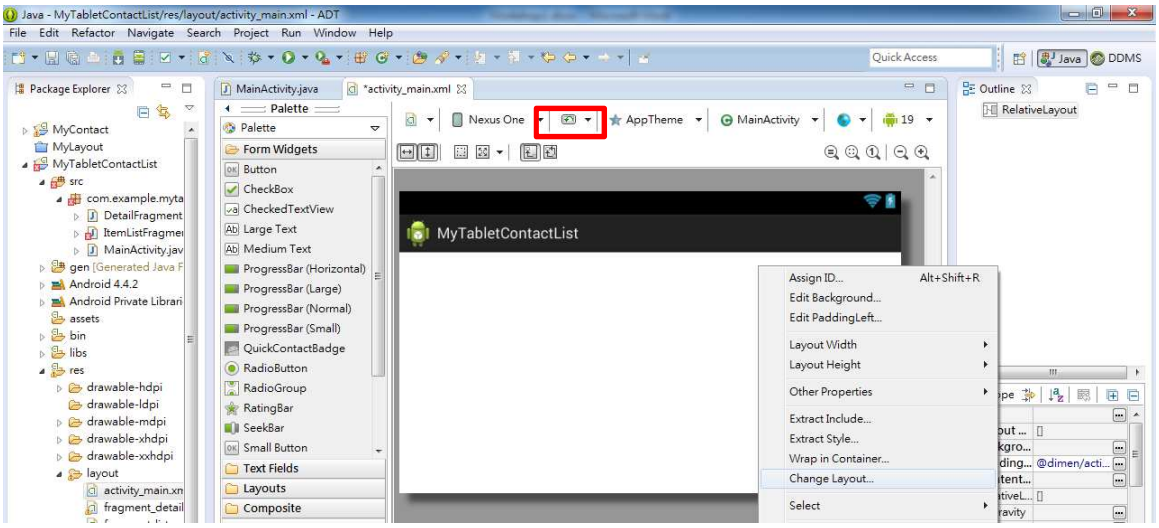
9. Modify "**ItemListFragment.java**" as follow:

```
package com.example.mytabletcontactlist;


import android.app.ListFragment;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
```

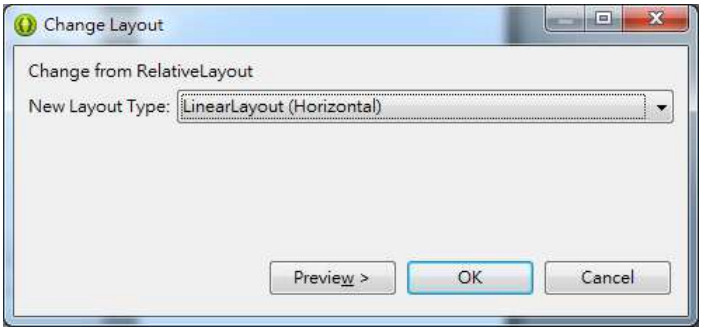
```
import android.widget.TextView;

public class ItemListFragment extends ListFragment {
    public SimpleCursorAdapter adapter;
    public final static String EXTRA_MESSAGE1 = "MyContactList.cid";
    public final static String EXTRA_MESSAGE2 = "MyContactList.Nickname";

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        // To retrieve the contact row that contains the user's profile
        ContentResolver contentResolver = getActivity().getContentResolver();

        // Queries the user dictionary based on selection criteria and returns results
        Cursor cursor = contentResolver.query(
                ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

        // Define a simple Adapter
        adapter = new SimpleCursorAdapter(getActivity(),
                    android.R.layout.simple_list_item_1,
                    cursor,
                    new String[] { ContactsContract.Contacts.DISPLAY_NAME },
                    new int[] { android.R.id.text1 });

        // Create the List Adapter
       setListAdapter(adapter);

       // Inflate the layout for this fragment
       return inflater.inflate(R.layout.fragment_list, container, false);
    }

    // Handle Item click event
    public void onListItemClick(ListView list, View view, int position, long id) {
        // Retrieve the user input
        Cursor cursor = (Cursor) adapter.getItem(position);

        // Get the Cursor Column Index
        int ColumnIndex = cursor.getColumnIndex(ContactsContract.Contacts._ID);
```

```
            // Convert the cursor Column Index into string
            String cid = cursor.getString(ColumnIndex);

            // Get the nickname
            String Nickname = (String) ((TextView) view).getText();

            // Obtain the Detail Fragment
            DetailFragment fragment = (DetailFragment)
                        getFragmentManager().findFragmentById(R.id.fragment2);
            if (fragment != null && fragment.isInLayout()) {
                // Display the detail in fragment
                fragment.setText(cid, Nickname);
            }
        }
}
```

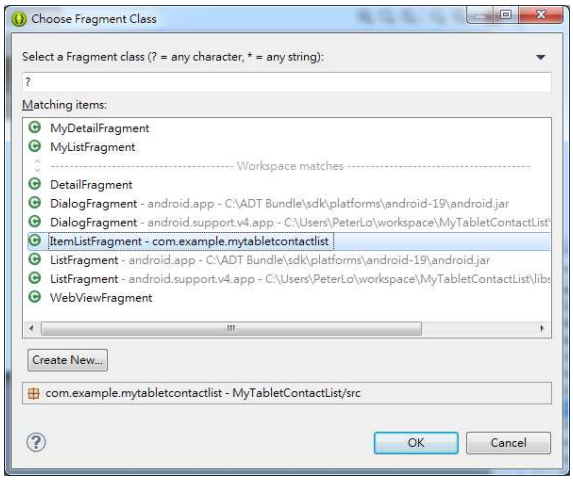
10. Switch the layout "**activity_main.xml**" to "**Landscape**" and remove the TextView "**Hello world!**". Then right click the layout and then select **Change Layout**.



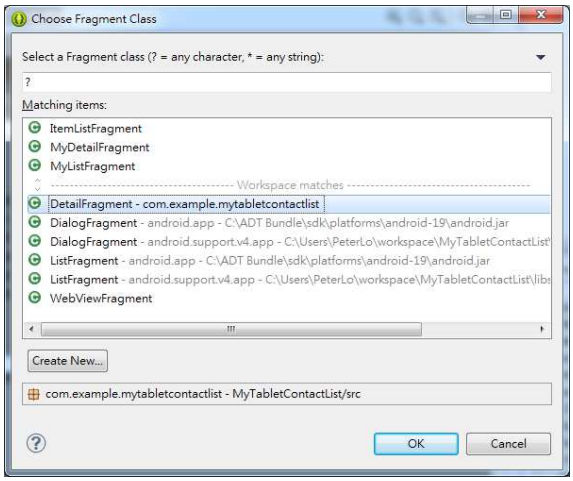
11. Select "**LinearLayout (Horizontal)**" in the "**Change Layout**" dialog.



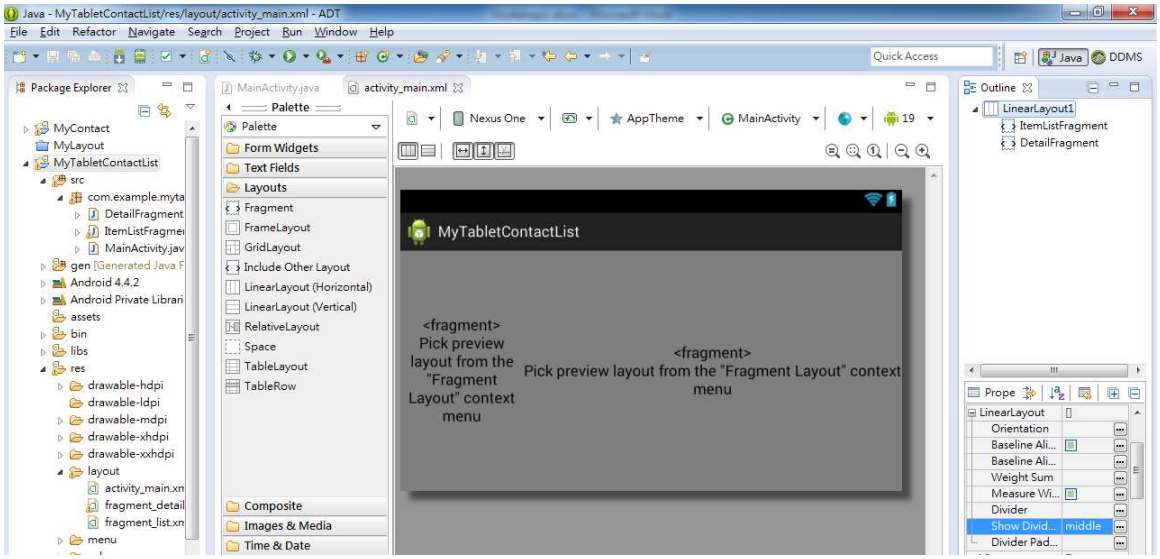
12. Drag a Fragment to the layout. Select "**ItemListFragment**" in the "**Choose Fragment Class**" dialog, and then press **[OK]** button.



13. Drag another Fragment to the layout. Select "**DetailFragment**" in the "**Choose Fragment Class**" dialog, and then press **[OK]** button.



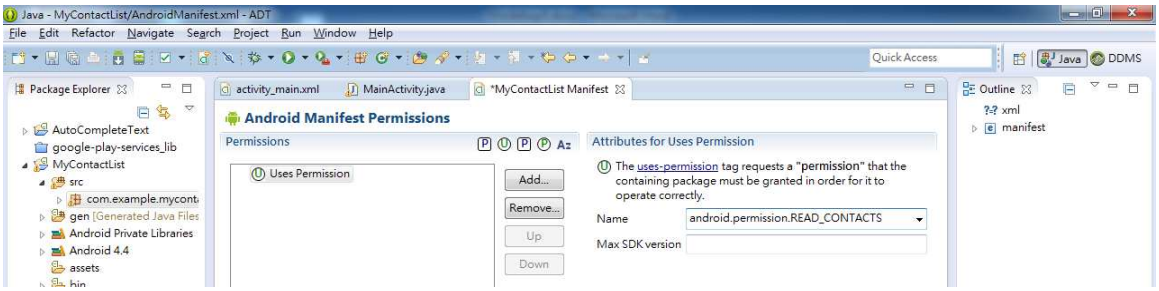
14. Set the properties of "**Show Dividers**" to "**middle**"



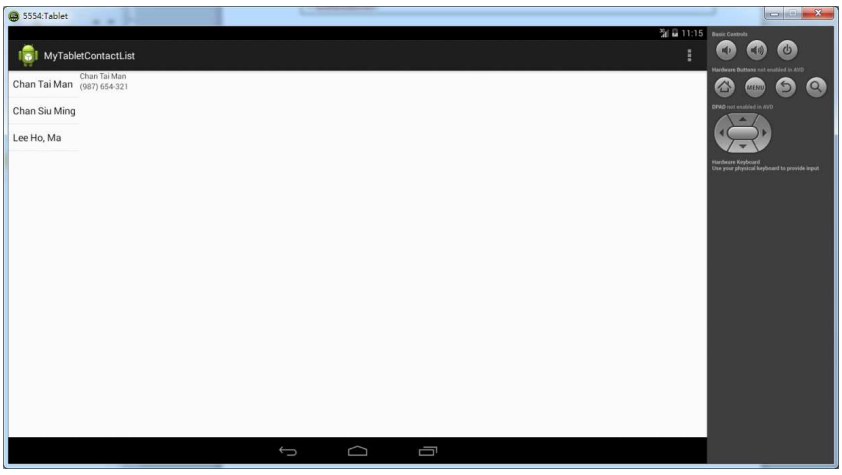
15. The XML for the layout will look like:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:showDividers="middle"
    tools:context=".MainActivity" >
    <fragment
        android:id="@+id/fragment1"
        android:name="com.example.mytabletcontactlist.ItemListFragment"
        android:layout_width="131dp"
        android:layout_height="match_parent" />
    <fragment
        android:id="@+id/fragment2"
        android:name="com.example.mytabletcontactlist.DetailFragment"
        android:layout_width="wrap_content"
        android:layout_height="match_parent" />
</LinearLayout>
```

16. Set the permission name as "**android.permission.READ_CONTACTS**" and save:



17. Save and execute the app in tablet AVD, you should able to display the detail by select the list:

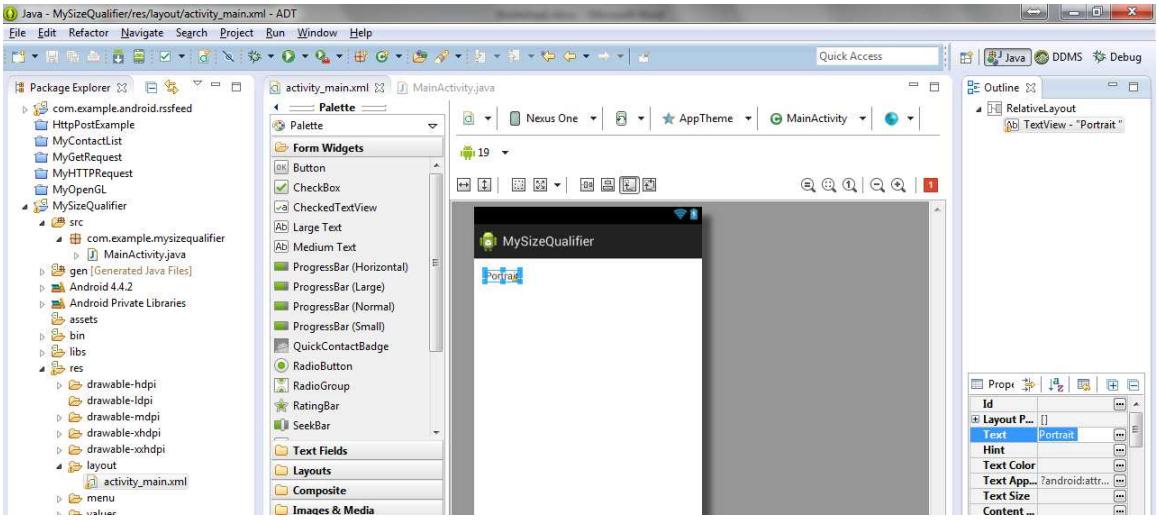


4. Handling Landscape and Portrait Layout

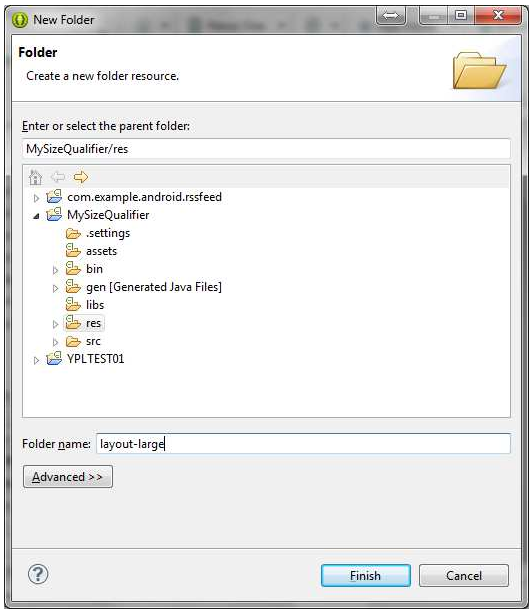
4.1 Using Size Qualifier

1. Create the Android application with the following attributes.
   - Application Name: **MySizeQualifier**
   - Project Name: **MySizeQualifier**
   - Package Name: **com.example.mysizequalifier**

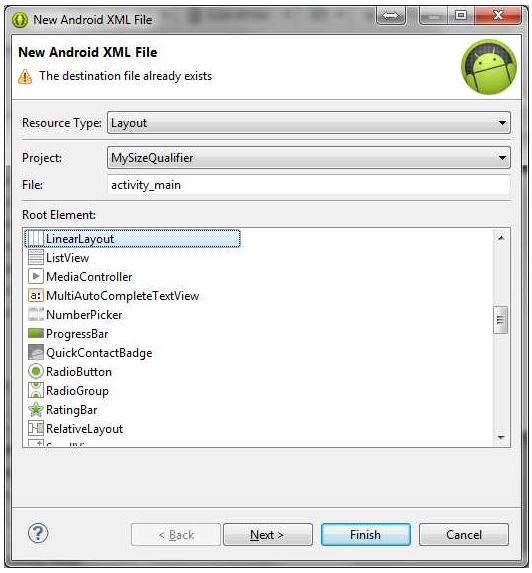
2. Change the text for the default TextView to "**Portrait**".



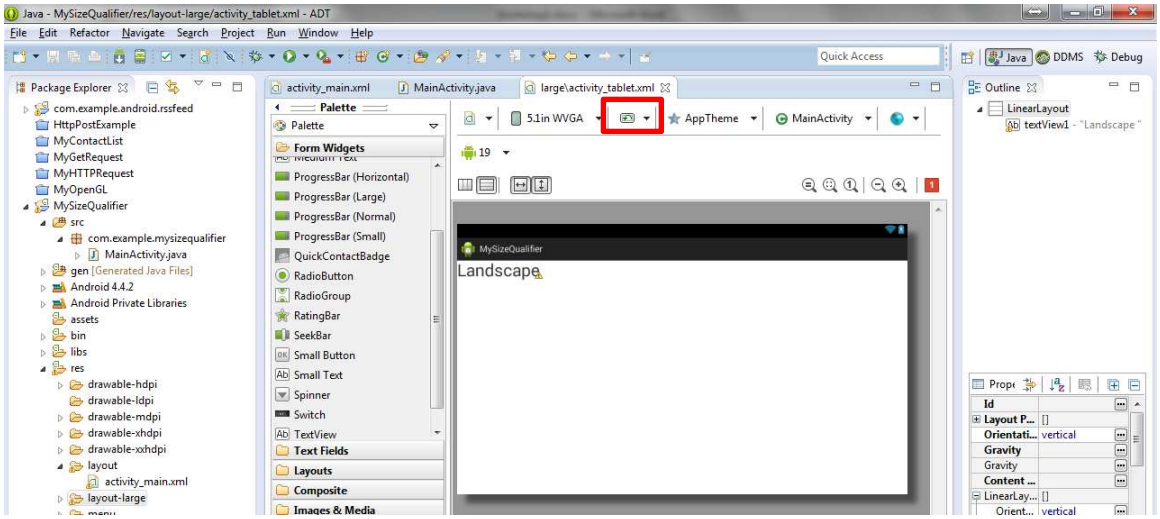
3. Right click the folder "**res**", and select **New ➔ Folder**. Input "**layout-large**" as the folder name and press **[Finish]** button. Notice the large qualifier in the directory name of the second layout. This layout will be selected on devices with screens classified as large (for example, 7" tablets and above). The other layout (without qualifiers) will be selected for smaller devices



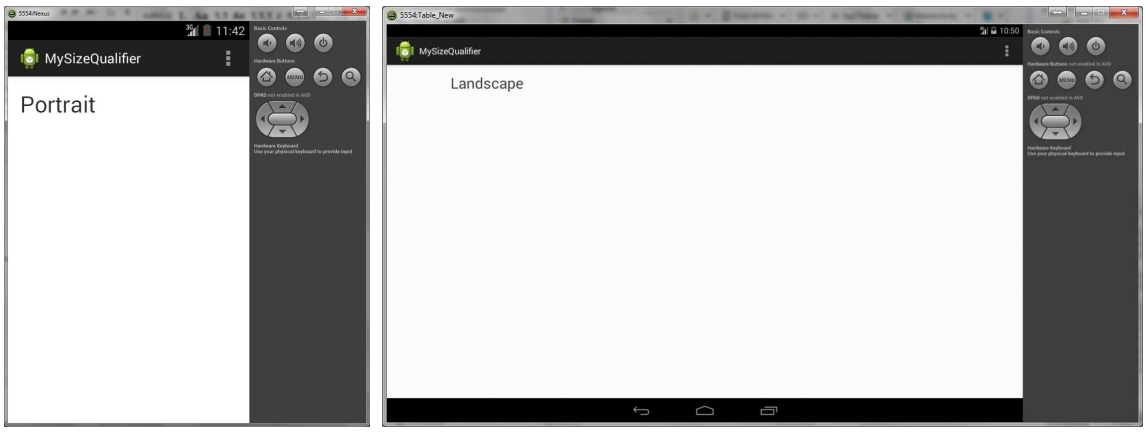
4. Right click the folder "**res/layout-large**", and select **New ➔ Android XML File**. Input "**activity_main**" as the file name and press **[Finish]** button.



5. Change the layout of the "**res/layout-large/activity_main.xml**" to landscape. Then drag a TextView to the layout and change the text to "**Landscape**".



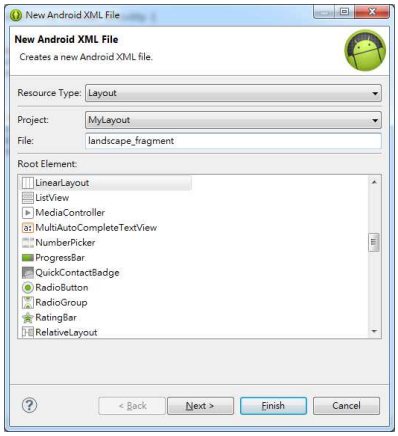
6. Save and execute the app in mobile and Tablet AVD, can you observe the different?



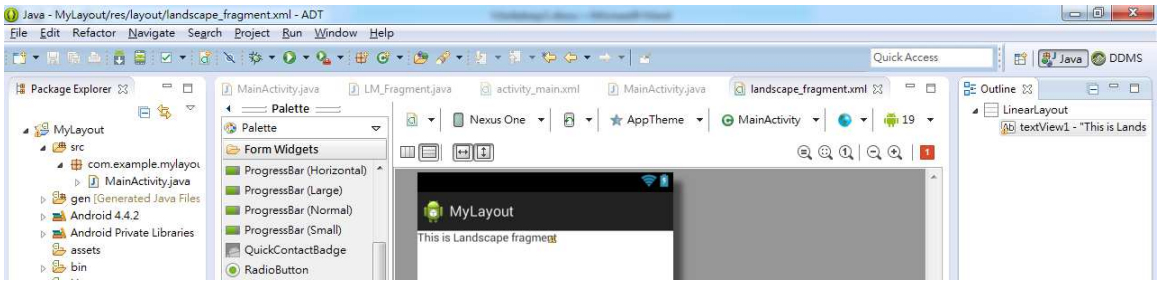
4.2 Building a Dynamic UI with Fragments

1. Create the Android application with the following attributes.
   - Application Name: **MyLayout**
   - Project Name: **MyLayout**
   - Package Name: **com.example.mylayout**

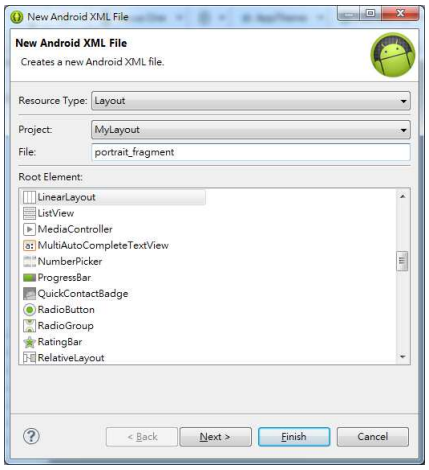
2. Right click the layout "**res/layout**", and select **New ➔ Android XML File**. Input "**landscape_fragment**" as the file name and press **[Finish]** button.



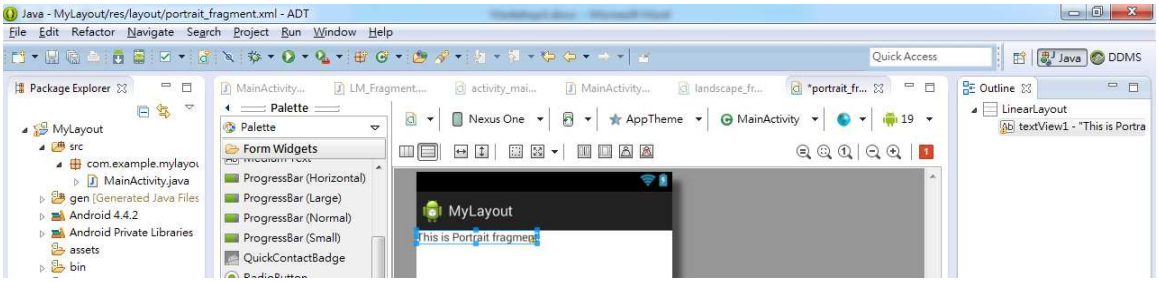
3. Change the layout to landscape**.** Then drag a TextView to the layout and change the text to "**This is Landscape fragment**".



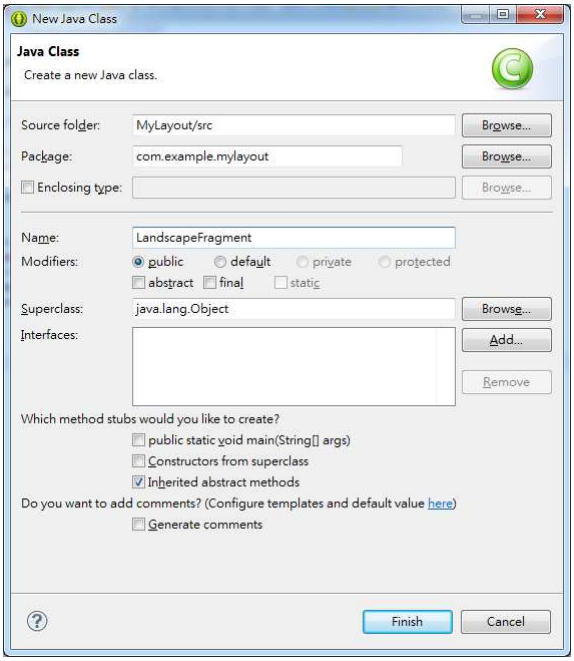
4. Right click the layout "**res/layout**", and select **New ➔ Android XML File**. Input "**portrait_fragment**" as the file name and press **[Finish]** button.



5. Drag a TextView to the layout and change the text to "**This is Portrait fragment**".



6. Right click the package **src/com.example.mylayout**, and select **New ➔ Class**. Input "**LandscapeFragment**" as the class name and press **[Finish]** button.

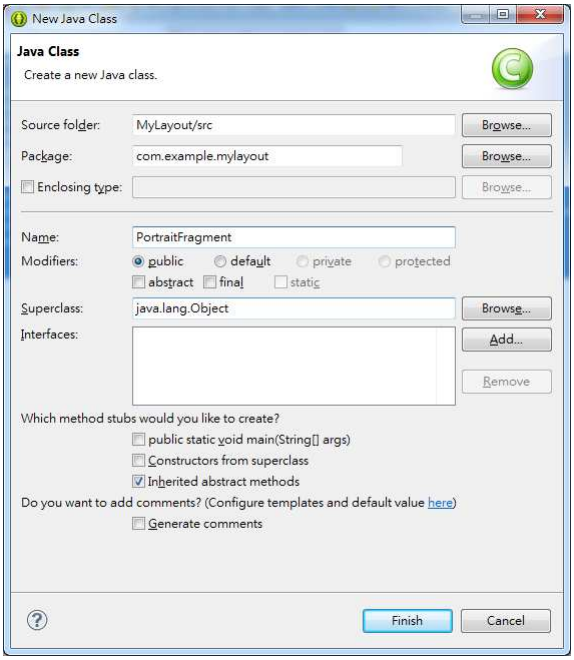


7. Modify the source code for the file "**LandscapeFragment.java**" as follow:

```
package com.example.mylayout;


import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;


public class LandscapeFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the landscape layout for this fragment
```

```
            return inflater.inflate(R.layout.landscape_fragment, container, false);
        }
}
```

8. Right click the package **src/com.example.mylayout**, and select **New ➔ Class**. Input "**PortraitFragment**" as the class name and press **[Finish]** button.



9. Modify the source code for the file "**PortraitFragment.java**" as follow:

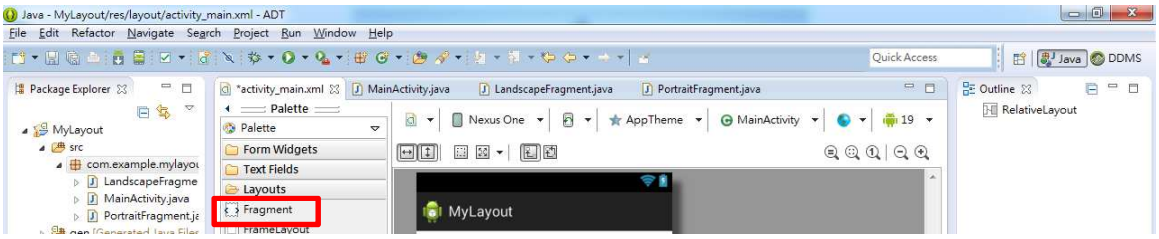
```
package com.example.mylayout;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PortraitFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the portrait layout for this fragment
        return inflater.inflate(R.layout.portrait_fragment, container, false);
    }
}
```

10. Modify the source code for the file "**MainActivity.java**" as follow:

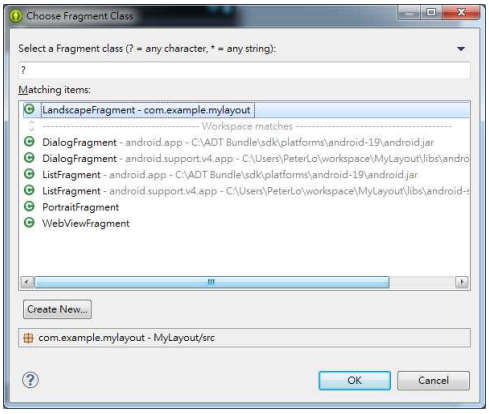
```
package com.example.mylayout;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
//      setContentView(R.layout.activity_main);

        // Obtain the layout configuration and setting
        Configuration config = getResources().getConfiguration();

        // Construct the Fragment Manager
        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
                            fragmentManager.beginTransaction();

        // Check the device orientation and act accordingly
        if (config.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            // Landscape mode of the device
            LandscapeFragment mLandscapeFragment = new LandscapeFragment();
           fragmentTransaction.replace(android.R.id.content, mLandscapeFragment);
        } else {
            // Portrait mode of the device
            PortraitFragment mPortraitFragment = new PortraitFragment();
            fragmentTransaction.replace(android.R.id.content, mPortraitFragment);
        }

        // Schedules a commit of this transaction
        fragmentTransaction.commit();
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

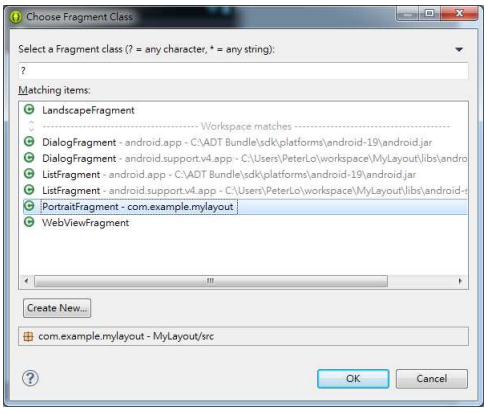
11. Remove the default TextView “Hello world” from the layout “**activity_main.xml**”. Then drag a fragment to the layout.



12. Select "**LandscapeFragment**" and press **[OK]** to confirm. Then change the width and height to “match_parent”.



13. Drag another fragment to the layout, select "**PortraitFragment**" and press **[OK]** to confirm. Then change the width and height to “**match_parent**”.



14. The XML of the layout "**activity_main.xml**" will look like:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/LandscapeFragment"
        android:name="com.example.mylayout.LandscapeFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <fragment
        android:id="@+id/PortraitFragment"
        android:name="com.example.mylayout.PortraitFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</RelativeLayout>
```

15. Save and execute the app in mobile and tablet AVD. If you execute it in your mobile, you can rotate it to see the change immediately.